PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-078942

(43)Date of publication of application : 24.03.1998

(51)Int.Cl. G06F 15/163
G06F 12/08

(21)Application number : 08-251000

(22)Date of filing : 02.09.1996

(71)Applicant : NEC CORP

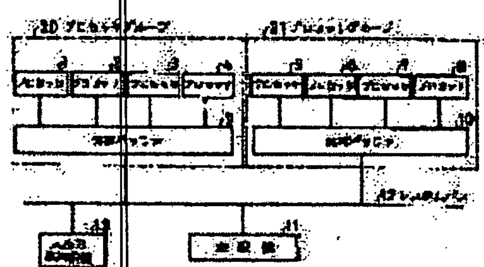
(72)Inventor : FUJIWARA YOSHIFUMI

(54) MULTIPROCESSOR SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To increase the degree of freedom of task assignment to processors by providing a shared buffer means which is used for buffering the data of a main storage via each processor of processor groups in each of these processor groups.

SOLUTION: The processor groups include a group 20 consisiting of processors 1 to 4 and a shared buffer 9 and a group 21 consisting of processors 5 to 8 and a shared buffer 10. It is not necessary that both groups 20 and 21 must have the same number of processors. For instance, one of both groups can have only one processor and the other group can have three or five processors. In other words, every processor group is just required to include one or more processors and a shared buffer which is shared by these processors.



LEGAL STATUS

[Date of request for examination]	02.09.1996
[Date of sending the examiner's decision of rejection]	27.06.2001
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	
[Date of registration]	
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（JP）　(12)公開特許公報(A)　(11)特許出願公開番号
特開平10−78942
(43)公開日 平成10年(1998)3月24日

(51)Int.Cl.⁶	識別記号	庁内整理番号	FI	技術表示箇所
G06F 15/163			G06F 15/16	320K
12/08		7623−5B	12/08	H

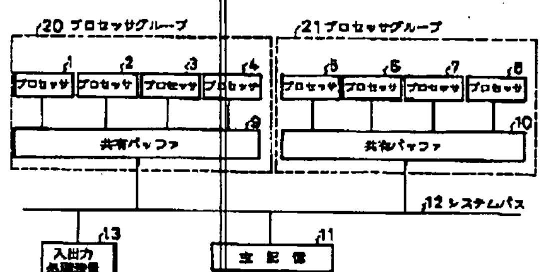
審査請求 有 請求項の数3 FD (全 8 頁)

(21)出願番号 特願平8−251000
(22)出願日 平成8年(1996)9月2日

(71)出願人 000004237
日本電気株式会社
東京都港区芝五丁目7番1号
(72)発明者 藤原 芳文
東京都港区芝五丁目7番1号 日本電気株式会社内
(74)代理人 弁理士 河原 純一

(54)【発明の名称】マルチプロセッサシステム

(57)【要約】
【課題】 プロセッサへのタスク割り当ての自由度を高め、キャッシュのミスヒットによる性能の低下およびメモリアクセス競合や負荷軽減を図る。
【解決手段】 複数個のプロセッサ1～8はプロセッサグループ20，21を単位として構成され、プロセッサグループ毎に主記憶11のデータをバッファする共有バッファ9，10を備える。プロセッサ管理手段はプロセッサグループに属するプロセッサを管理し、キュー手段はプロセッサグループ毎に設定された待ち状態キュー群からなり、タスク割り当て手段はエンキュー時にキュー手段のどの待ち状態キューにタスクを割り当てるかを制御し、検出手段はディスパッチ要求の要求元プロセッサ番号からプロセッサグループを検出し、ローカルディスパッチ指示手段は再ディスパッチング時にローカルタスクかグローバルタスクかを判断し、ディスパッチング手段はその結果に基づき選択するタスクを決定する。



【特許請求の範囲】

【請求項１】　複数個のプロセッサから構成され、タスクを単位として該プロセッサにディスパッチし処理を実行するマルチプロセッサシステムにおいて、

前記複数個のプロセッサは該プロセッサの幾つかをまとめたプロセッサグループを単位として該プロセッサグループを１つ以上合わせることにより構成されるものとし、該プロセッサグループ毎に該プロセッサグループの各プロセッサによって主記憶のデータをバッファするために共有される共有バッファ手段を備えることを特徴とするマルチプロセッサシステム。

【請求項２】　前記請求項１のマルチプロセッサシステムにおいて、該プロセッサグループに属するプロセッサを管理するプロセッサ管理手段と、該プロセッサグループ毎に設置された待ち状態キュー群からなるキュー手段と、タスク生成によるエンキュー時に該タスクを前記キュー手段のどの待ち状態キューに割り当てるかを制御するタスク割り当て手段とを備えるマルチプロセッサシステム。

【請求項３】　前記請求項２のマルチプロセッサシステムにおいて、ディスパッチ要求の要求元プロセッサ番号から該プロセッサが属するプロセッサグループを検出する検出手段と、前記タスクが再ディスパッチング時に同一プロセッサグループ内のプロセッサのみにディスパッチされるタスクかそれ以外のプロセッサグループにもディスパッチ可能なタスクかを判断するローカルディスパッチ指示手段とを備え、その結果に基づき選択するタスクを決定するディスパッチング手段を有するマルチプロセッサシステム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明はマルチプロセッサシステムに関し、特に高並列プロセッサの効率的なプロセッサ構成とそのプロセッサ構成に対する効率的なタスクディスパッチング手法に関する。

【０００２】

【従来の技術】従来のマルチプロセッサシステムにおける各プロセッサへのタスク割り当て方式は、例えば、特開平３－４４７４２号公報の第２図に示されているように、実行待ち状態のタスクのタスク制御ブロック（以下、ＴＣＢと略記する）リストから予め定められた１つのＴＣＢを選択し、そのＴＣＢが初めてプロセッサに割り当てられたものであれば即座に処理を実行する。また、ＴＣＢが初めてプロセッサに割り当てられたものでない場合には、前回プロセッサに割り当てられた時からの経過時間が一定時間以上であるか、前回処理されたプロセッサと同一であるかを判断する。この場合、判定がイエスであればタスクのプロセッサへの割り当てを実施するが、ノーであればＴＣＢリストから予め定められた手続きに従い選択されていない１つのＴＣＢを選択し、

新たに選択されたＴＣＢについて、上記の処理を繰り返し、判定を実施する。選択されていないＴＣＢがなければ、ＴＣＢリストから予め定められた手続きに従い１つのＴＣＢを選択し、処理を実行するようになっている。

【０００３】

【発明が解決しようとする課題】上述した従来の技術の第１の問題点は、タスクが再ディスパッチングされるプロセッサが前回と同一のプロセッサのみと固定されていることである。その理由は、同一プロセッサに再ディスパッチングすることによって得られる効果はキャッシュにある従前のデータを再利用できることにあるためであり、キャッシュ効果が得られない場合には同一プロセッサへの再ディスパッチングによる効果はないからである。

【０００４】第２の問題点は、処理ユニットの構成においてキャッシュが個々のプロセッサに独立して構成されていることである。その理由は、本来キャッシュは個々のプロセッサからのメモリアクセス要求に対して高速なデータ供給手段であるとユニプロセッサ時代から位置づけられていたため、ディスパッチング効果を図ることを目的としてキャッシュを複数のプロセッサで共有するといった考えが無かったためである。

【０００５】本発明の目的は、タスクのプロセッサへの再ディスパッチング時に、同一のプロセッサのみを対象とするのではなく、プロセッサをグループ化することにより、該プロセッサが含まれているプロセッサグループ内であればどのプロセッサでも再ディスパッチングによる効果が得られる構成とし、プロセッサへのタスク割り当ての自由度を高めるとともに、キャッシュのミスヒットによる性能の低下およびメモリアクセス競合やそれに伴う負荷の軽減を図り、高並列プロセッサの処理効率を向上させるマルチプロセッサシステムを提供することにある。

【０００６】

【課題を解決するための手段】本発明のマルチプロセッサシステムは、複数個のプロセッサから構成され、タスクを単位として該プロセッサにディスパッチし処理を実行するマルチプロセッサシステムにおいて、前記複数個のプロセッサは該プロセッサの幾つかをまとめたプロセッサグループを単位として該プロセッサグループを１つ以上合わせることにより構成されるものとし、該プロセッサグループ毎に該プロセッサグループの各プロセッサによって主記憶のデータをバッファするために共有される共有バッファ手段を備え、該プロセッサグループに属するプロセッサを管理するプロセッサ管理手段と、該プロセッサグループ毎に設定された待ち状態キュー群からなるキュー手段と、タスク生成によるエンキュー時に前記キュー手段のどの待ち状態キューにタスクを割り当てるかを制御するタスク割り当て手段と、ディスパッチ要求の要求元プロセッサ番号から該プロセッサが属するプ

ロセッサグループを検出する検出手段と、前記タスクが再ディスパッチング時に同一プロセッサグループ内のプロセッサのみにディスパッチされるタスクかそれ以外のプロセッサグループにもディスパッチ可能なタスクかを判断するローカルディスパッチ指示手段と、その結果に基づき選択するタスクを決定するディスパッチング手段とを有する。

【０００７】

【発明の実施の形態】次に、本発明について図面を参照して詳細に説明する。

【０００８】図１は、本発明の一実施の形態に係るマルチプロセッサシステムの全体構成を示すブロック図である。本実施の形態に係るマルチプロセッサシステムは、共有バッファ（２次キャッシュ）９を共有してプロセッサグループ２０としてグルーピングされたプロセッサ１～４と、共有バッファ（２次キャッシュ）１０を共有してプロセッサグループ２１としてグルーピングされたプロセッサ５～８と、主記憶１１と、システムバス１２と、入出力処理装置１３とから、その主要部が構成されている。なお、図１には、主記憶１１のアクセスパスを主とした接続態様を示しており、プロセッサ１～８とシステムバス１２とを直接接続するバス等も存在するものである。

【０００９】プロセッサ１～８は、処理の単位であるタスクを実行する処理機構である。

【００１０】共有バッファ９および１０は、プロセッサ１～４またはプロセッサ５～８の各々が共通にアクセス可能で、主記憶部１１にあるデータを保持するための高速バッファ機構である。

【００１１】主記憶部１１は、プロセッサ１～８によって使用されるデータ等を保持する記憶機構である。

【００１２】システムバス１２は、共有バッファ９および１０と、主記憶１１と、入出力処理装置１３とを接続するバスである。

【００１３】入出力処理装置１３は、磁気ディスク装置等の外部装置へのアクセス処理制御するための装置である。

【００１４】本実施の形態に係るマルチプロセッサシステムにおけるプロセッサグループは、プロセッサ１～４および共有バッファ９で構成されているプロセッサグループ２０と、プロセッサ５～８および共有バッファ１０で構成されているプロセッサグループ２１との２グループであるものとする。

【００１５】ここで、プロセッサグループ内のプロセッサ数は同数である必要はなく、例えば、１プロセッサのみのプロセッサグループと、３プロセッサのプロセッサグループや５プロセッサのプロセッサグループとであってもかまわない。すなわち、１つ以上のプロセッサとそのプロセッサによって共有可能な共有バッファとで構成されていればよい。

【００１６】図２を参照すると、オペレーティングシステム（ＯＳ）のタスク管理プログラム中の待ち状態キュー管理手段（図示せず）が、生成されたタスクを各プロセッサグループの待ち状態キューにエンキューするまでの処理は、タスクタイプ設定ステップ５１と、プロセッサグループ決定ステップ５２と、待ち状態キューへの登録ステップ５３とから構成されている。

【００１７】図３を参照すると、ＯＳのタスク管理プログラム中のディスパッチャ（図示せず）の処理は、要求元プロセッサ番号検出ステップ６１と、プロセッサグループ検出ステップ６２と、待ち状態キュー検索ステップ６３と、待ち状態タスク有無判定ステップ６４と、別プロセッサグループ待ち状態キュー検索ステップ６５と、待ち状態タスク有無判定ステップ６６と、グローバルタスク判定ステップ６７と、プロセッサアサインステップ６８と、未検索プロセッサグループ待ち状態キュー有無判定ステップ６９とからなる。

【００１８】マルチプロセッサシステムにおいては、同時に複数のプロセッサで処理ができるようにタスク毎にＴＣＢが設けられており、タスクが何らかの事象（例えば、入出力割り込み、タイムスライス等）に伴う割り込み等により処理が中断した場合に、その処理が再開できるようにＴＣＢにその情報を格納している。

【００１９】図４を参照すると、タスクの生成に伴い生成されるＴＣＢは、タスクタイプと、プロセッサグループ番号と、タスクを再開するための情報とからなる。タスクタイプには、グローバルタスクとローカルタスクとの２種類がある。グローバルタスクは、プロセッサにて処理が実行される場合、個々のプロセッサグループに固定されることなくどのプロセッサグループにおいても実行可能なタスクであり、ローカルタスクは一旦プロセッサグループが指定されると以後はそのプロセッサグループのみにディスパッチされるタスクである。このタスクタイプは、ジョブ制御言語（ＪＣＬ）によって指定される。ＯＳ等のシステム系のタスクに関しては、規定タイプを予めシステム生成（ＳＧ）等において指定しておくことも可能である。タスクタイプの指定がされていない場合には、グローバルタスクとしての指定が自動的に行われる。

【００２０】図５を参照すると、プロセッサグループ管理テーブルは、プロセッサグループ番号と、プロセッサグループ毎にそのプロセッサグループに属するプロセッサ番号と、そのプロセッサグループに対応する待ち状態キューの先頭キューのアドレスを示すポインタとから構成されている。プロセッサグループに属するプロセッサは、そのプロセッサに対応するプロセッサ番号を示すビットに“１”が設定されることによって表示される。

【００２１】図６を参照すると、待ち状態キューにエンキュー／デキューされる各キューは、タスクタイプと、その他の制御情報と、ＴＣＢへのポインタと、次のキュ

ーへのポインタとからなる。

【0022】次に、このように構成された本実施の形態に係るマルチプロセッサシステムの動作について説明する。

【0023】まず、タスクを各プロセッサグループの待ち状態キューにエンキューするキューイング処理について、図2のフローチャートを参照しながら説明する。

【0024】タスクが生成されると、待ち状態キュー管理手段は、JCLによって指定された、あるいは自動的に決定されたタスクのタスクタイプを図4に示すTCB上に登録するとともに、図6に示すキュー上にも登録する（ステップ51）。

【0025】ステップ51でタスクタイプが設定されると、待ち状態キュー管理手段は、プロセッサグループを決定する（ステップ52）。このプロセッサグループへの割り振りは、JCLで指定することによって強制的に制御する方法と、ラウンドロビン等の既存の負荷分散手法によって自動的に割り振ることとが可能である。この両者の選択は、JCLによるプロセッサグループの指定がなかった場合に、自動的に後者のアルゴリズムによって指定される。本実施の形態では、後者に関しては、ラウンドロビンを前提に処理されるものとする。なお、OS等のシステム系のタスクに関しては実行するプロセッサグループを予めSG等において指定しておくことも可能である。

【0026】ステップ52でプロセッサグループが決定されると、待ち状態キュー管理手段は、そのプロセッサグループに対応した待ち状態キューにタスクをエンキューする（ステップ53）。これにより、キューイング処理が完了する。

【0027】次に、プロセッサグループ20および21の個々のプロセッサ1～8から次に実行するタスクを要求された場合の動作について、図3のフローチャートを参照しながら説明する。

【0028】プロセッサからタスクのディスパッチング要求があると、ディスパッチャは、まず、要求元プロセッサ番号を検出する（ステップ61）。プロセッサ番号を検出すると、ディスパッチャは、そのプロセッサの属するプロセッサグループを割り出し、そのプロセッサグループに対応する待ち状態キューの先頭を示すポインタ（アドレス）を入手する（ステップ62）。この一連の作業は、図5に示したプロセッサグループ管理テーブルを検索することにより行われる。

【0029】プロセッサグループに対応する待ち状態キューの先頭を示すポインタを入手すると、ディスパッチャは、そのプロセッサグループに対応する待ち状態キューに対し待ち状態のタスクがあるかどうかの検索を開始する（ステップ63）。本実施の形態では、待ち状態のタスクの有無の判定は、キューを示すポインタ先のデータエリアの全ビットが“1”である場合に待ち状態のタスクがないと判断する。

【0030】待ち状態のタスクが存在する場合には（ステップ64でイエス）、ディスパッチャは、ポインタ先のキューが示すタスクを選択し（デキュー）、選択したタスクをプロセッサにアサインして（ステップ68）、ディスパッチング処理を完了する。

【0031】タスクをディスパッチングされたプロセッサは、プロセッサグループ毎に共有バッファを備えているため、タスクが再ディスパッチングされたプロセッサが前回と異なったプロセッサであっても、そのプロセッサが前回のプロセッサと同一プロセッサグループ内のプロセッサであるので、プロセッサ内部のキャッシュ（1次キャッシュ）でヒットしなくても、共有バッファによるキャッシュ効果で主記憶11へのアクセスを抑えられるために性能低下を軽減できる。

【0032】一方、ステップ64で要求元プロセッサ番号のプロセッサが属するプロセッサグループ（以下、自プロセッサグループという）の待ち状態キューに待ち状態のタスクが存在しなかった場合、すなわちキューを示すポインタ先のデータエリアの全ビットが“1”である場合、ディスパッチャは、別のプロセッサグループの待ち状態キューを検索し（ステップ65）、この待ち状態キューに対して待ち状態のタスクが存在しないかどうかを検索する（ステップ66）。この場合に、次のプロセッサグループの選択は、現行のプロセッサグループ番号＋1のプロセッサグループ番号を選択する。そして、プロセッサグループ番号が最大値になると、最小値のプロセッサグループ番号のプロセッサグループに移り、自プロセッサグループまで待ち状態キューの検索を繰り返す。図1に示す本実施の形態に係るマルチプロセッサシステムでは、プロセッサグループがプロセッサグループ20および21の2グループしかないので、現行のプロセッサグループが20であれば21に、プロセッサグループが21であれば20に移る。

【0033】ここで、プロセッサグループがプロセッサグループG1～G4までの4グループであったとし、自プロセッサグループがG3であったすると、ディスパッチャは、自プロセッサグループG3の待ち状態キューからプロセッサグループG4→G1→G2の待ち状態キューへと待ち状態のタスクを見つけるまで検索を繰り返し、最終的に自プロセッサグループG3の待ち状態キューに戻るまで待ち状態のタスクがなければ全てのプロセッサグループの待ち状態キューを検索したこととなる。

【0034】ステップ66で待ち状態のタスクがあった場合には、ディスパッチャは、選択された待ち状態のタスクのタスクタイプがグローバルタスクかローカルタスクかを判断する（ステップ67）。これは、図6に示されるキュー上のタスクタイプをみることによって判断する。

【0035】タスクタイプがグローバルタスクでなくロ

ーカルタスクである場合には、ディスパッチャは、別の待ち状態のタスクを選択するために、ステップ６６に制御を戻す。

【００３６】タスクタイプがグローバルタスクであれば、ディスパッチャは、その待ち状態のタスクを選択し（デキュー）、選択したタスクをプロセッサにアサインして（ステップ６８）、ディスパッチング処理を完了する。この場合、ディスパッチャは、選択された待ち状態のタスクのＴＣＢのプロセッサグループを、ディスパッチャを起動させた要求元プロセッサ番号のプロセッサの属するプロセッサグループに変更する。

【００３７】ステップ６６で別のプロセッサグループの待ち状態キューにも待ち状態のタスクがなければ、ディスパッチャは、さらに未検索のプロセッサグループがあるかどうかを判定し（ステップ６９）、未検索のプロセッサグループがあれば、ステップ６５に制御を戻し、ステップ６５以降の処理を繰り返す。

【００３８】全てのプロセッサグループの待ち状態キューを検索して、待ち状態のタスクがなければ（ステップ６９でノー）、ディスパッチャは、ディスパッチング処理を終了する。

【００３９】

【発明の効果】以上説明したように、本発明の第１の効果は、キャッシュ効果を生かしたプロセッサへのタスクの再ディスパッチングにおいて、前回実行したプロセッサのみが対象となるのではなく、そのプロセッサを含むプロセッサグループ内のプロセッサであれば全て対象にすることが可能であるということである。これにより、処理のターンアラウンドタイムの短縮をはかれ、プロセッサの使用効率を高めて、ＭＰ（マルチプロセッサ）係数の向上に大きく寄与することができる。また、これにより、キャッシュ効果を高めメモリアクセス頻度を軽減し、ひいてはメモリ競合やシステムバスビジィの軽減を図れる。その理由は、第１に、システム構成を１つ以上のプロセッサとそのプロセッサの全てでデータ共有が可能な共有バッファとからなるプロセッサグループによってシステムが構成されているために、プロセッサグループ内でのデータ使用効率を高められるからである。第２に、待ち状態キューをプロセッサグループ単位に設定することにより、ディスパッチャの処理負荷を高めることなくグループディスパッチング処理を可能としたことである。

【００４０】また、本発明の第２の効果は、プロセッサ障害時におけるプロセッサ数のデグレード時においてもキャッシュ効果を生かしたディスパッチング処理が可能なことである。その理由は、従来技術で示されているような、前回実行したプロセッサのみを再ディスパッチングの対象とするのではなく、プロセッサグループ内のプロセッサであればどのプロセッサでも再ディスパッチングの対象となるグループディスパッチング処理を行っているためである。

【００４１】さらに、本発明の第３の効果は、ローカルタスクおよびグローバルタスクのタスクタイプを設定したことにより、負荷分散が図られることである。その理由は、自プロセッサグループ内で待ち状態のタスクが無い場合には、別のプロセッサグループのグローバルタスクを処理することが可能であるためである。

【００４２】さらにまた、本発明の第４の効果は、特定用途に応じたプロセッサの分散処理が可能なことである。その理由は、ローカルタスクとしてプロセッサグループを指定することにより、プロセッサグループ単位にタスクのディスパッチングを可能とするためである。

【図面の簡単な説明】

【図１】本発明の一実施の形態に係るマルチプロセッサシステムの全体構成を例示する図である。

【図２】ＯＳのタスク管理プログラム中の待ち状態キュー管理手段がタスクを各プロセッサグループの待ち状態キューにエンキューするまでの処理手順を示すフローチャートである。

【図３】ＯＳのタスク管理プログラム中のディスパッチャがタスクをプロセッサにディスパッチングするための処理手順を示すフローチャートである。

【図４】待ち状態キュー管理手段が使用するＴＣＢの構成を示す図である。

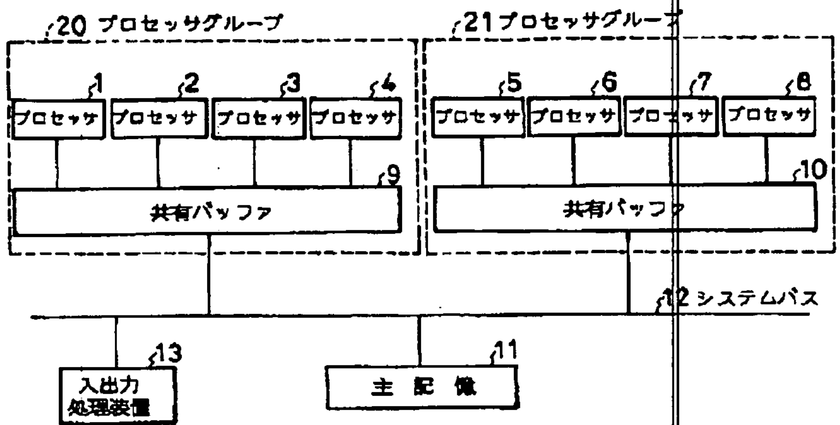
【図５】ディスパッチャが使用するプロセッサグループ管理テーブルの内容を示す図である。

【図６】待ち状態キュー管理手段およびディスパッチャが使用する待ち状態キューを構成する各キューの内容を示す図である。

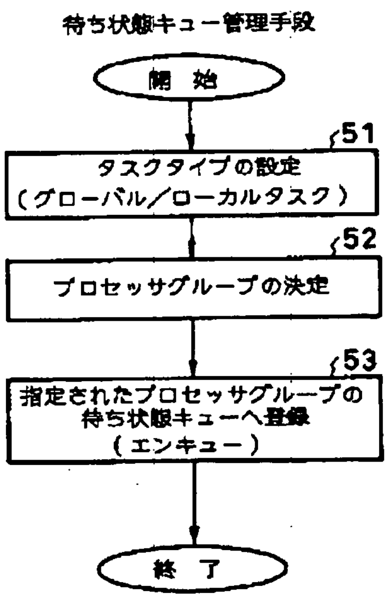
【符号の説明】

１～８ プロセッサ
９，１０ 共有バッファ
１１ 主記憶
１２ システムバス
１３ 入出力処理装置
２０，２１ プロセッサグループ
５１ タスクタイプ設定ステップ
５２ プロセッサグループ決定ステップ
５３ 待ち状態キューへの登録ステップ
６１ 要求元プロセッサ番号検出ステップ
６２ プロセッサグループ検出ステップ
６３ 待ち状態キュー検索ステップ
６４ 待ち状態タスク有無判定ステップ
６５ 別プロセッサグループ待ち状態キュー検索ステップ
６６ 待ち状態タスク有無判定ステップ
６７ グローバルタスク判定ステップ
６８ プロセッサアサインステップ
６９ 未検索プロセッサグループ待ち状態キュー有無判定ステップ

【図1】



【図2】



【図4】

TCB

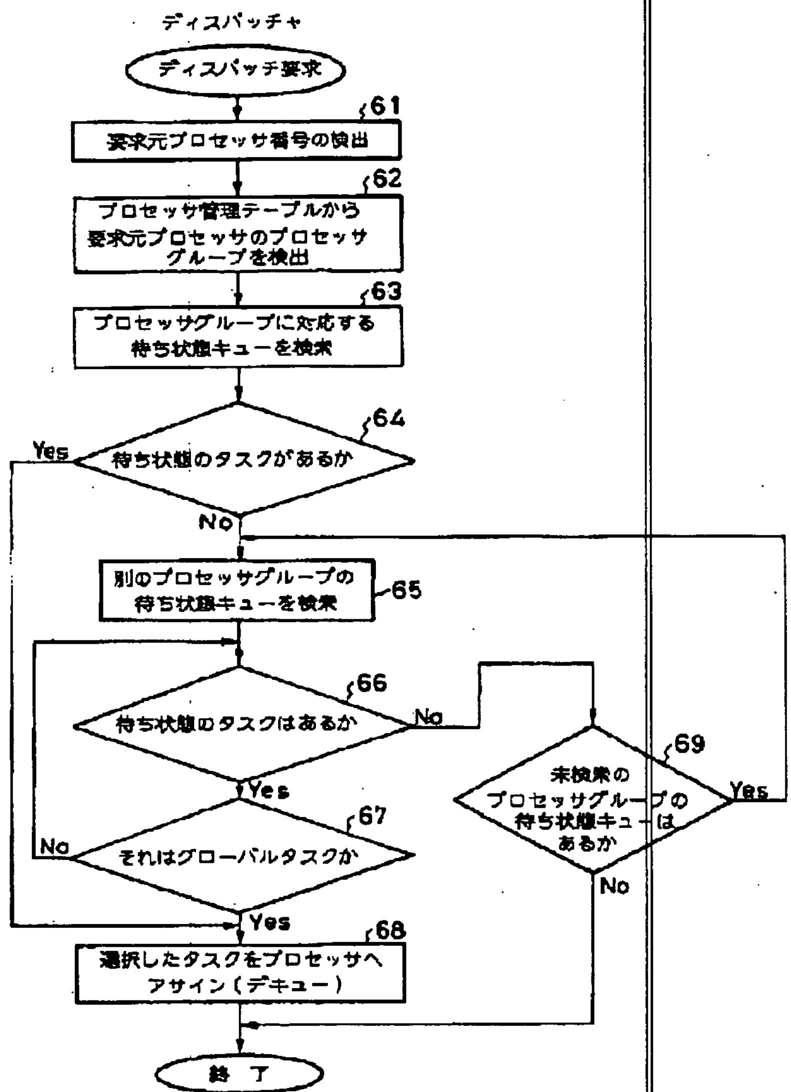
タスクタイプ
プロセッサグループ番号
タスクを再開するための情報

【図6】

キュー

タスクタイプ	その他の制御情報	TCBへのポインタ	次のキューへのポインタ

【図3】


ディスパッチャ
ディスパッチ要求
61
要求元プロセッサ番号の検出
62
プロセッサ管理テーブルから要求元プロセッサのプロセッサグループを検出
63
プロセッサグループに対応する待ち状態キューを検索
64
待ち状態のタスクがあるか
Yes
No
別のプロセッサグループの待ち状態キューを検索
65
66
待ち状態のタスクはあるか
No
Yes
67
それはグローバルタスクか
No
Yes
69
未検索のプロセッサグループの待ち状態キューはあるか
Yes
No
68
選択したタスクをプロセッサへアサイン（デキュー）
終了

【図5】

プロセッサグループ管理テーブル

プロセッサグループ番号	プロセッサ番号 (1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	待ち状態キューの先頭を示すポインタ
1	1	1	1	1	0	0	0	0	プロセッサグループ1の待ち状態キューのポインタ
2	0	0	0	0	1	1	1	1	プロセッサグループ2の待ち状態キューのポインタ

CLAIMS

[Claim(s)]
[Claim 1] In the multiprocessor system which consists of two or more processors, dispatches to this processor by making a task into a unit, and performs processing It shall be constituted by said two or more processors' making a unit the processor group who summarized these some of processors, and doubling this processor group one or more. The multiprocessor system characterized by having the shared buffer means shared in order to carry out the buffer of the data of a primary storage by each processor of this processor group for this every processor group.
[Claim 2] A multiprocessor system equipped with the processor management tool which manages the processor belonging to this processor group in the multiprocessor system of said claim 1, the queue means which consists of a waiting state queue group installed for this every processor group, and a task quota means to control which waiting state queue of said queue means this task is assigned at the time of the ENQ by creation of task.
[Claim 3] The multiprocessor system which has a dispatching means determine the task which is equipped with a detection means detect the processor group to whom this processor belongs from the requiring agency processor number of a dispatch demand in the multiprocessor system of said claim 2, and a local dispatch directions means judge the task by which said task is dispatched only to the processor in the same processor group at the time of re-dispatching, or the task which can be dispatched also to the other processor group, and is chosen based on the result.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] This invention relates to the efficient task dispatching technique for the efficient processor configuration and its processor configuration of especially a high juxtaposition processor about a multiprocessor system.
[0002]
[Description of the Prior Art] The task quota method to each processor in the conventional multiprocessor system will perform processing immediately, if one TCB defined beforehand is chosen from the task control block (it is hereafter written as TCB) list of tasks of an activation waiting state and the TCB is assigned to a processor for the first time as shown in drawing 2 of JP,3-44742,A. Moreover, when TCB is not what was assigned to the processor for the first time, and assigned to a processor last time, it judges whether the elapsed time of a from is beyond fixed time amount, or it is the same as that of the processor processed last time. In this case, although assignment to the processor of a task will be carried out if a judgment is yes, TCB which chose one TCB which is not chosen from the TCB list according to the procedure defined beforehand when it was a no, and was newly chosen is judged by repeating the above-mentioned processing. If there is no TCB which is not chosen, one TCB will be chosen from a TCB list according to the procedure defined beforehand, and processing will be performed.
[0003]
[Problem(s) to be Solved by the Invention] The 1st trouble of a Prior art mentioned above is that the processor to which the re-dispatching of the task is carried out is being fixed only with the same processor as last time. It is because it is in the ability of the effectiveness acquired by carrying out the re-dispatching of the reason to the same processor to reuse data old [in a cache], and is because there is no effectiveness by the re-dispatching to the same processor when the cache effectiveness is not acquired.
[0004] The 2nd trouble is that the cache is independently constituted by each processor in the configuration of a processing unit. Originally, since the cache was positioned from the uniprocessor age to the memory access demand from each processor as it is a high-speed data supply means, the reason is because there was no idea that two or more processors shared a cache for the purpose of planning the dispatching effectiveness.
[0005] The object of this invention is not aimed only at the same processor at the time of the re-dispatching to the processor of a task. If it is in the processor group in whom this processor is contained by carrying out grouping of the processor, the effectiveness according to re-dispatching also in which processor will consider as the configuration from which it is obtained. While raising the degree of freedom of the task assignment to a processor, relief of the load accompanying the performance degradation and memory access contention by the mistake hit of a cache, or it is aimed at, and it is in offering the multiprocessor system which raises the processing effectiveness of a high parallel processor.
[0006]
[Means for Solving the Problem] In the multiprocessor system which the multiprocessor system of this invention consists of two or more processors, dispatches to this processor by making a task into a unit, and performs processing It shall be constituted by said two or more processors' making a unit the processor group who summarized these some of processors, and doubling this processor group one or more. It has the shared buffer means shared in order to carry out the buffer of the data of a primary storage by each processor of this processor group for this every processor group. The processor management tool which manages the processor belonging to this processor group, The queue means which consists of a waiting state queue group set up for this every processor group, A task quota means to control which waiting state queue of said queue means a task is assigned at the time of the ENQ by creation of task, A detection means to detect the processor group to whom this processor belongs from the requiring agency processor number of a

dispatch demand, A local dispatch directions means to judge the task by which said task is dispatched only to the processor in the same processor group at the time of re-dispatching, or the task which can be dispatched also to the other processor group, It has a dispatching means to determine the task chosen based on the result.

[0007]

[Embodiment of the Invention] Next, this invention is explained to a detail with reference to a drawing.

[0008] Drawing 1 is the block diagram showing the whole multiprocessor system configuration concerning the gestalt of 1 operation of this invention. The body consists of the processors 1-4 by which the multiprocessor system concerning the gestalt of this operation shared the shared buffer (level 2 cache) 9, and grouping was carried out as a processor group 20, the processors 5-8 by which shared the shared buffer (level 2 cache) 10, and grouping was carried out as a processor group 21, a primary storage 11, a system bus 12, and an input/output processor 13. In addition, the connection mode mainly concerned with the access path of a primary storage 11 is shown in drawing 1 , and the pass which carries out direct continuation of processors 1-8 and the system bus 12 exists in it.

[0009] Processors 1-8 are processors which perform the task which is the unit of processing.

[0010] Shared buffers 9 and 10 have accessible each of processors 1-4 or processors 5-8 in common, and are the high-speed buffer styles for holding the data in a primary storage 11.

[0011] A primary storage 11 is the storage holding the data used by processors 1-8.

[0012] A system bus 12 is a bus which connects shared buffers 9 and 10, a primary storage 11, and an input/output processor 13.

[0013] An input/output processor 13 is equipment for [to external devices, such as a magnetic disk drive,] carrying out access processing control.

[0014] The processor groups in the multiprocessor system concerning the gestalt of this operation shall be two groups of the processor group 20 who consists of processors 1-4 and a shared buffer 9, and the processor group 21 who consists of processors 5-8 and a shared buffer 10.

[0015] Here, the number of processors in a processor group does not need to be the same number, for example, may be with the processor group of only one processor, and the processor group of three processors and the processor group of five processors. Namely, what is necessary is just to consist of shared buffers sharable [with one or more processors and the processor of those].

[0016] Reference of drawing 2 constitutes processing until the waiting state queue management tool (not shown) in the task administrator of an operating system (OS) carries out the ENQ of the attached task to each processor group's waiting state queue from a task type setting-out step 51, a processor group decision step 52, and a registration step 53 to a waiting state queue.

[0017] When drawing 3 is referred to, processing of the dispatcher in the task administrator of OS (not shown) The requiring agency processor number detection step 61 and the processor group detection step 62, The waiting state queue retrieval step 63 and the waiting state task existence judging step 64, It consists of another processor group waiting state queue retrieval step 65, the waiting state task existence judging step 66, the global task judging step 67, a processor assignment step 68, and a non-searched processor group waiting state queue existence judging step 69.

[0018] In a multiprocessor system, TCB is prepared for every task so that processing can be simultaneously done in two or more processors, and when processing is interrupted by interruption accompanying some events (for example, I/O interruption, a time slice, etc.) in a task etc., the information is stored in TCB so that the processing can be resumed.

[0019] If drawing 4 is referred to, TCB generated in connection with a creation of task will consist of a task type, the processor group number, and information for resuming a task. There are two kinds such as a global task and a local task of task types. A global task is a task which can be performed in every processor group, without being fixed to each processor group when processing is performed in a processor, and once, as for a local task, a processor group is specified, it is the task dispatched only to the processor group henceforth. This task type is specified by job control language (JCL). It is also possible to specify a convention type in system generation (SG) etc. about the task of system systems, such as OS, beforehand. When task type assignment is not carried out, assignment as a global task is performed automatically.

[0020] Reference of drawing 5 constitutes the processor group managed table from the processor group number, a processor number which belongs to the processor group for every processor group, and a pointer in which the address of the head queue of the waiting state queue corresponding to the processor group is shown. The processor belonging to a processor group is displayed by setting "1" as the bit which shows the processor number corresponding to the processor.

[0021] If drawing 6 is referred to, an ENQ / each queue by which a dequeue is carried out will turn into a waiting state queue from a task type, other control information, the pointer to TCB, and the pointer to the following queue.
[0022] Next, actuation of the multiprocessor system concerning the gestalt of this operation constituted in this way is explained.
[0023] First, the queuing processing which carries out the ENQ of the task to each processor group's waiting state queue is explained, referring to the flow chart of drawing 2 .
[0024] If a task is generated, a waiting state queue management tool will be registered also on the queue shown in drawing 6 while registering it on TCB which shows the task type of the task which was specified by JCL or was determined automatically to drawing 4 (step 51).
[0025] If a task type is set up at step 51, a waiting state queue management tool will determine a processor group (step 52). Assignment in this processor group can be automatically assigned by specifying by JCL by the approach of controlling compulsorily, and the existing load-distribution technique, such as a round robin. These both selection is automatically specified by the latter algorithm, when there is no assignment of the processor group by JCL. With the gestalt of this operation, about the latter, it shall be processed on the assumption that a round robin. In addition, it is also possible to specify beforehand the processor group who performs about the task of system systems, such as OS, in SG etc.
[0026] If a processor group is determined at step 52, a waiting state queue management tool will carry out the ENQ of the task to the waiting state queue corresponding to the processor group (step 53). Thereby, queuing processing is completed.
[0027] Next, actuation when the task performed next from each processors 1-8 of the processor groups 20 and 21 is required is explained, referring to the flow chart of drawing 3 .
[0028] If there is a dispatching demand of a task from a processor, as for a dispatcher, a requiring agency processor number will be detected first (step 61). If a processor number is detected, a dispatcher will deduce the processor group to whom the processor belongs, and will receive the pointer (address) in which the head of the waiting state queue corresponding to the processor group is shown (step 62). This activity of a series of is done by searching the processor group managed table shown in drawing 5 .
[0029] If the pointer in which the head of the waiting state queue corresponding to a processor group is shown comes to hand, a dispatcher will start retrieval of whether there is any task of a waiting state to the waiting state queue corresponding to the processor group (step 63). With the gestalt of this operation, it is judged that the judgment of the existence of the task of a waiting state does not have the task of a waiting state when all the bits of the data area of the pointer point which shows a queue are "1."
[0030] When the task of a waiting state exists, at the (step 64, yes) and a dispatcher choose the task which the queue of the pointer point shows (dequeue), assign the selected task to a processor (step 68), and complete dispatching processing.
[0031] Since the processor is the last processor and a processor in the same processor group even if the processor to which the re-dispatching of the task was carried out is a different processor from last time, since it has the shared buffer for every processor group, even if it does not hit in the cache inside a processor (level 1 cache), since the processor by which dispatching was carried out in the task can suppress access to a primary storage 11 by the cache effectiveness by the shared buffer, it can mitigate degradation.
[0032] When the task of a waiting state does not exist in a processor group's (henceforth an intraprocessor group) waiting state queue with which the processor of a requiring agency processor number belongs at step 64 on the other hand (i.e., when all the bits of the data area of the pointer point which shows a queue are "1"), a dispatcher searches another processor group's waiting state queue (step 65), and searches whether the task of a waiting state exists to this waiting state queue (step 66). In this case, the next processor group's selection chooses the processor group number of the present processor group number +1. And if the processor group number becomes maximum, it will move to the processor group of the processor group number of the minimum value, and even an intraprocessor group will repeat retrieval of a waiting state queue. In the multiprocessor system concerning the gestalt of this operation shown in drawing 1 , since there are only only two the processor groups' 20 and 21 groups' processor groups, if the present processor group is 20, if a processor group is 21, it will move to 21 20.
[0033] it presupposed that processor groups were four groups to the processor groups G1-G4 here, and the intraprocessor group was G3 -- a dispatcher, if it carries out Retrieval is repeated until it finds the task of a waiting state from the waiting state queue of intraprocessor group G3 to the waiting state queue of processor group G4 ->G1 ->G2. If there is no task of a waiting state until it returns to the waiting state queue of intraprocessor group G3 eventually, it

will mean searching all processor groups' waiting state queue.
[0034] When there is a task of a waiting state at step 66, the task type of the task of a waiting state with which the dispatcher was chosen judges a global task or a local task (step 67). This is judged by seeing the task type on the queue shown in drawing 6 .
[0035] When a task type is not a global task but a local task, a dispatcher returns control to step 66, in order to choose the task of another waiting state.
[0036] If a task type is a global task, a dispatcher chooses the task of the waiting state (dequeue), will assign the selected task to a processor (step 68), and will complete dispatching processing. In this case, a dispatcher is changed into the processor group to whom the processor of the started requiring agency processor number belongs a dispatcher in the processor group of TCB of the task of the selected waiting state.
[0037] If it judges whether a dispatcher has a non-searched processor group further at step 66 if there is no task of a waiting state also in another processor group's waiting state queue (step 69) and there is a non-searched processor group, control will be returned to step 65 and the processing after step 65 will be repeated.
[0038] If all processor groups' waiting state queue is searched and there is no task of a waiting state (it is no at step 69), a dispatcher will end dispatching processing.
[0039]
[Effect of the Invention] As explained above, in the re-dispatching of the task to the processor which employed the cache effectiveness efficiently, only the processor performed last time is not applicable, but if the 1st effectiveness of this invention is a processor in the processor group containing the processor, I hear that it can make all into an object, and there is. By this, compaction of the turn around time of processing can be aimed at, the utilization ratio of a processor can be raised, and it can contribute to improvement in MP (multiprocessor) multiplier greatly. Moreover, by this, the cache effectiveness is heightened, and memory access frequency is mitigated, as a result memory contention and system bus busy relief can be aimed at. The reason is that the data utilization ratio within a processor group is raised since the system is constituted [1st] from all one or more processor and processor of those in the system configuration by the processor group who consists of a shared buffer in which data sharing is possible. It is having enabled group dispatching processing by setting a waiting state queue as the 2nd per processor group, without raising the processing load of a dispatcher.
[0040] Moreover, the 2nd effectiveness of this invention is that the dispatching processing in which the cache effectiveness was efficiently employed at the time of the degradation of the number of processors at the time of a processor failure is possible. The reason does not set only the processor performed last time [as shown with the conventional technique] as the object of re-dispatching, but if it is a processor in a processor group, it is because group dispatching processing in which every processor is set as the object of re-dispatching is performed.
[0041] Furthermore, the 3rd effectiveness of this invention is that a load distribution is planned by having set up the task type of a local task and a global task. The reason is because it is possible to process another processor group's global task, when there is no task of a waiting state within an intraprocessor group.
[0042] The 4th effectiveness of this invention is that distributed processing of the processor according to a specified use is possible further again. The reason is for making the dispatching of a task possible per processor group by specifying a processor group as a local task.

[Translation done.]

TECHNICAL FIELD

[Field of the Invention] This invention relates to the efficient task dispatching technique for the efficient processor configuration and its processor configuration of especially a high juxtaposition processor about a multiprocessor system.

[Translation done.]

PRIOR ART

[Description of the Prior Art] The task quota method to each processor in the conventional multiprocessor system will perform processing immediately, if one TCB defined beforehand is chosen from the task control block (it is hereafter written as TCB) list of tasks of an activation waiting state and the TCB is assigned to a processor for the first time as shown in drawing 2 of JP,3-44742,A. Moreover, when TCB is not what was assigned to the processor for the first time, and assigned to a processor last time, it judges whether the elapsed time of a from is beyond fixed time amount, or it is the same as that of the processor processed last time. In this case, although assignment to the processor of a task will be carried out if a judgment is yes, TCB which chose one TCB which is not chosen from the TCB list according to the procedure defined beforehand when it was a no, and was newly chosen is judged by repeating the above-mentioned processing. If there is no TCB which is not chosen, one TCB will be chosen from a TCB list according to the procedure defined beforehand, and processing will be performed.

[Translation done.]

EFFECT OF THE INVENTION

[Effect of the Invention] As explained above, in the re-dispatching of the task to the processor which employed the cache effectiveness efficiently, only the processor performed last time is not applicable, but if the 1st effectiveness of this invention is a processor in the processor group containing the processor, I hear that it can make all into an object, and there is. By this, compaction of the turn around time of processing can be aimed at, the utilization ratio of a processor can be raised, and it can contribute to improvement in MP (multiprocessor) multiplier greatly. Moreover, by this, the cache effectiveness is heightened, and memory access frequency is mitigated, as a result memory contention and system bus busy relief can be aimed at. The reason is that the data utilization ratio within a processor group is raised since the system is constituted [1st] from all one or more processor and processor of those in the system configuration by the processor group who consists of a shared buffer in which data sharing is possible. It is having enabled group dispatching processing by setting a waiting state queue as the 2nd per processor group, without raising the processing load of a dispatcher.
[0040] Moreover, the 2nd effectiveness of this invention is that the dispatching processing in which the cache effectiveness was efficiently employed at the time of the degradation of the number of processors at the time of a processor failure is possible. The reason does not set only the processor performed last time [as shown with the conventional technique] as the object of re-dispatching, but if it is a processor in a processor group, it is because group dispatching processing in which every processor is set as the object of re-dispatching is performed.
[0041] Furthermore, the 3rd effectiveness of this invention is that a load distribution is planned by having set up the task type of a local task and a global task. The reason is because it is possible to process another processor group's global task, when there is no task of a waiting state within an intraprocessor group.
[0042] The 4th effectiveness of this invention is that distributed processing of the processor according to a specified use is possible further again. The reason is for making the dispatching of a task possible per processor group by specifying a processor group as a local task.

[Translation done.]

TECHNICAL PROBLEM

[Problem(s) to be Solved by the Invention] The 1st trouble of a Prior art mentioned above is that the processor to which the re-dispatching of the task is carried out is being fixed only with the same processor as last time. It is because it is in the ability of the effectiveness acquired by carrying out the re-dispatching of the reason to the same processor to reuse data old [in a cache], and is because there is no effectiveness by the re-dispatching to the same processor when the cache effectiveness is not acquired.
[0004] The 2nd trouble is that the cache is independently constituted by each processor in the configuration of a processing unit. Originally, since the cache was positioned from the uniprocessor age to the memory access demand from each processor as it is a high-speed data supply means, the reason is because there was no idea that two or more processors shared a cache for the purpose of planning the dispatching effectiveness.
[0005] The object of this invention is not aimed only at the same processor at the time of the re-dispatching to the processor of a task. If it is in the processor group in whom this processor is contained by carrying out grouping of the processor, the effectiveness according to re-dispatching also in which processor will consider as the configuration from which it is obtained. While raising the degree of freedom of the task assignment to a processor, relief of the load accompanying the performance degradation and memory access contention by the mistake hit of a cache, or it is aimed at, and it is in offering the multiprocessor system which raises the processing effectiveness of a high parallel processor.

[Translation done.]

MEANS

[Means for Solving the Problem] In the multiprocessor system which the multiprocessor system of this invention consists of two or more processors, dispatches to this processor by making a task into a unit, and performs processing It shall be constituted by said two or more processors' making a unit the processor group who summarized these some of processors, and doubling this processor group one or more. It has the shared buffer means shared in order to carry out the buffer of the data of a primary storage by each processor of this processor group for this every processor group. The processor management tool which manages the processor belonging to this processor group, The queue means which consists of a waiting state queue group set up for this every processor group, A task quota means to control which waiting state queue of said queue means a task is assigned at the time of the ENQ by creation of task, A detection means to detect the processor group to whom this processor belongs from the requiring agency processor number of a dispatch demand, A local dispatch directions means to judge the task by which said task is dispatched only to the processor in the same processor group at the time of re-dispatching, or the task which can be dispatched also to the other processor group, It has a dispatching means to determine the task chosen based on the result.
[0007]
[Embodiment of the Invention] Next, this invention is explained to a detail with reference to a drawing.
[0008] Drawing 1 is the block diagram showing the whole multiprocessor system configuration concerning the gestalt of 1 operation of this invention. The body consists of the processors 1-4 by which the multiprocessor system concerning the gestalt of this operation shared the shared buffer (level 2 cache) 9, and grouping was carried out as a processor group 20, the processors 5-8 by which shared the shared buffer (level 2 cache) 10, and grouping was carried out as a processor group 21, a primary storage 11, a system bus 12, and an input/output processor 13. In addition, the connection mode mainly concerned with the access path of a primary storage 11 is shown in drawing 1 , and the pass which carries out direct continuation of processors 1-8 and the system bus 12 exists in it.
[0009] Processors 1-8 are processors which perform the task which is the unit of processing.
[0010] Shared buffers 9 and 10 have accessible each of processors 1-4 or processors 5-8 in common, and are the high-speed buffer styles for holding the data in a primary storage 11.
[0011] A primary storage 11 is the storage holding the data used by processors 1-8.
[0012] A system bus 12 is a bus which connects shared buffers 9 and 10, a primary storage 11, and an input/output processor 13.
[0013] An input/output processor 13 is equipment for [to external devices, such as a magnetic disk drive,] carrying out access processing control.
[0014] The processor groups in the multiprocessor system concerning the gestalt of this operation shall be two groups of the processor group 20 who consists of processors 1-4 and a shared buffer 9, and the processor group 21 who consists of processors 5-8 and a shared buffer 10.
[0015] Here, the number of processors in a processor group does not need to be the same number, for example, may be with the processor group of only one processor, and the processor group of three processors and the processor group of five processors. Namely, what is necessary is just to consist of shared buffers sharable [with one or more processors and the processor of those].
[0016] Reference of drawing 2 constitutes processing until the waiting state queue management tool (not shown) in the task administrator of an operating system (OS) carries out the ENQ of the attached task to each processor group's waiting state queue from a task type setting-out step 51, a processor group decision step 52, and a registration step 53 to a waiting state queue.
[0017] When drawing 3 is referred to, processing of the dispatcher in the task administrator of OS (not shown) The requiring agency processor number detection step 61 and the processor group detection step 62, The waiting state

queue retrieval step 63 and the waiting state task existence judging step 64, It consists of another processor group waiting state queue retrieval step 65, the waiting state task existence judging step 66, the global task judging step 67, a processor assignment step 68, and a non-searched processor group waiting state queue existence judging step 69.
[0018] In a multiprocessor system, TCB is prepared for every task so that processing can be simultaneously done in two or more processors, and when processing is interrupted by interruption accompanying some events (for example, I/O interruption, a time slice, etc.) in a task etc., the information is stored in TCB so that the processing can be resumed.
[0019] If drawing 4 is referred to, TCB generated in connection with a creation of task will consist of a task type, the processor group number, and information for resuming a task. There are two kinds such as a global task and a local task of task types. A global task is a task which can be performed in every processor group, without being fixed to each processor group when processing is performed in a processor, and once, as for a local task, a processor group is specified, it is the task dispatched only to the processor group henceforth. This task type is specified by job control language (JCL). It is also possible to specify a convention type in system generation (SG) etc. about the task of system systems, such as OS, beforehand. When task type assignment is not carried out, assignment as a global task is performed automatically.
[0020] Reference of drawing 5 constitutes the processor group managed table from the processor group number, a processor number which belongs to the processor group for every processor group, and a pointer in which the address of the head queue of the waiting state queue corresponding to the processor group is shown. The processor belonging to a processor group is displayed by setting "1" as the bit which shows the processor number corresponding to the processor.
[0021] If drawing 6 is referred to, an ENQ / each queue by which a dequeue is carried out will turn into a waiting state queue from a task type, other control information, the pointer to TCB, and the pointer to the following queue.
[0022] Next, actuation of the multiprocessor system concerning the gestalt of this operation constituted in this way is explained.
[0023] First, the queuing processing which carries out the ENQ of the task to each processor group's waiting state queue is explained, referring to the flow chart of drawing 2 .
[0024] If a task is generated, a waiting state queue management tool will be registered also on the queue shown in drawing 6 while registering it on TCB which shows the task type of the task which was specified by JCL or was determined automatically to drawing 4 (step 51).
[0025] If a task type is set up at step 51, a waiting state queue management tool will determine a processor group (step 52). Assignment in this processor group can be automatically assigned by specifying by JCL by the approach of controlling compulsorily, and the existing load-distribution technique, such as a round robin. These both selection is automatically specified by the latter algorithm, when there is no assignment of the processor group by JCL. With the gestalt of this operation, about the latter, it shall be processed on the assumption that a round robin. In addition, it is also possible to specify beforehand the processor group who performs about the task of system systems, such as OS, in SG etc.
[0026] If a processor group is determined at step 52, a waiting state queue management tool will carry out the ENQ of the task to the waiting state queue corresponding to the processor group (step 53). Thereby, queuing processing is completed.
[0027] Next, actuation when the task performed next from each processors 1-8 of the processor groups 20 and 21 is required is explained, referring to the flow chart of drawing 3 .
[0028] If there is a dispatching demand of a task from a processor, as for a dispatcher, a requiring agency processor number will be detected first (step 61). If a processor number is detected, a dispatcher will deduce the processor group to whom the processor belongs, and will receive the pointer (address) in which the head of the waiting state queue corresponding to the processor group is shown (step 62). This activity of a series of is done by searching the processor group managed table shown in drawing 5 .
[0029] If the pointer in which the head of the waiting state queue corresponding to a processor group is shown comes to hand, a dispatcher will start retrieval of whether there is any task of a waiting state to the waiting state queue corresponding to the processor group (step 63). With the gestalt of this operation, it is judged that the judgment of the existence of the task of a waiting state does not have the task of a waiting state when all the bits of the data area of the pointer point which shows a queue are "1."
[0030] When the task of a waiting state exists, at the (step 64, yes) and a dispatcher choose the task which the queue of the pointer point shows (dequeue), assign the selected task to a processor (step 68), and complete dispatching

processing.

[0031] Since the processor is the last processor and a processor in the same processor group even if the processor to which the re-dispatching of the task was carried out is a different processor from last time, since it has the shared buffer for every processor group, even if it does not hit in the cache inside a processor (level 1 cache), since the processor by which dispatching was carried out in the task can suppress access to a primary storage 11 by the cache effectiveness by the shared buffer, it can mitigate degradation.

[0032] When the task of a waiting state does not exist in a processor group's (henceforth an intraprocessor group) waiting state queue with which the processor of a requiring agency processor number belongs at step 64 on the other hand (i.e., when all the bits of the data area of the pointer point which shows a queue are "1"), a dispatcher searches another processor group's waiting state queue (step 65), and searches whether the task of a waiting state exists to this waiting state queue (step 66). In this case, the next processor group's selection chooses the processor group number of the present processor group number +1. And if the processor group number becomes maximum, it will move to the processor group of the processor group number of the minimum value, and even an intraprocessor group will repeat retrieval of a waiting state queue. In the multiprocessor system concerning the gestalt of this operation shown in drawing 1 , since there are only only two the processor groups' 20 and 21 groups' processor groups, if the present processor group is 20, if a processor group is 21, it will move to 21 20.

[0033] it presupposed that processor groups were four groups to the processor groups G1-G4 here, and the intraprocessor group was G3 -- a dispatcher, if it carries out Retrieval is repeated until it finds the task of a waiting state from the waiting state queue of intraprocessor group G3 to the waiting state queue of processor group G4 ->G1 ->G2. If there is no task of a waiting state until it returns to the waiting state queue of intraprocessor group G3 eventually, it will mean searching all processor groups' waiting state queue.

[0034] When there is a task of a waiting state at step 66, the task type of the task of a waiting state with which the dispatcher was chosen judges a global task or a local task (step 67). This is judged by seeing the task type on the queue shown in drawing 6 .

[0035] When a task type is not a global task but a local task, a dispatcher returns control to step 66, in order to choose the task of another waiting state.

[0036] If a task type is a global task, a dispatcher chooses the task of the waiting state (dequeue), will assign the selected task to a processor (step 68), and will complete dispatching processing. In this case, a dispatcher is changed into the processor group to whom the processor of the started requiring agency processor number belongs a dispatcher in the processor group of TCB of the task of the selected waiting state.

[0037] If it judges whether a dispatcher has a non-searched processor group further at step 66 if there is no task of a waiting state also in another processor group's waiting state queue (step 69) and there is a non-searched processor group, control will be returned to step 65 and the processing after step 65 will be repeated.

[0038] If all processor groups' waiting state queue is searched and there is no task of a waiting state (it is no at step 69), a dispatcher will end dispatching processing.

[Translation done.]

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] It is drawing which illustrates the whole multiprocessor system configuration concerning the gestalt of 1 operation of this invention.
[Drawing 2] It is the flow chart which shows procedure until the waiting state queue management tool in the task administrator of OS carries out the ENQ of the task to each processor group's waiting state queue.
[Drawing 3] It is the flow chart which shows procedure for the dispatcher in the task administrator of OS to carry out the dispatching of the task to a processor.
[Drawing 4] It is drawing showing the configuration of TCB which a waiting state queue management tool uses.
[Drawing 5] It is drawing showing the content of the processor group managed table which a dispatcher uses.
[Drawing 6] It is drawing showing the content of each queue which constitutes the waiting state queue which a waiting state queue management tool and a dispatcher use.
[Description of Notations]
1-8 Processor
9 Ten Shared buffer
11 Primary Storage
12 System Bus
13 Input/output Processor
20 21 Processor group
51 Task Type Setting-Out Step
52 Processor Group Decision Step
53 Registration Step to Waiting State Queue
61 Requiring Agency Processor Number Detection Step
62 Processor Group Detection Step
63 Waiting State Queue Retrieval Step
64 Waiting State Task Existence Judging Step
65 Another Processor Group Waiting State Queue Retrieval Step
66 Waiting State Task Existence Judging Step
67 Global Task Judging Step
68 Processor Assignment Step
69 Non-Searched Processor Group Waiting State Queue Existence Judging Step

---

[Translation done.]